МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №6_2

по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Поиск образца в тексте

Выполнил студент группы ИКБО-42-23 Голев С.С.

Принял ассистент Муравьёва Е.А.

Москва 2024

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ

Освоить приёмы реализации алгоритмов поиска образца в тексте..

1.УСЛОВИЯ ЗАДАЧ

Задание 1:

Устная часть.

Задание 2:

Разработайте приложения в соответствии с заданиями в индивидуальном варианте.

В отчёте в разделе «Математическая модель решения (описание алгоритма)» разобрать алгоритм поиска на примере.

Подсчитать количество сравнений для успешного поиска первого вхождения образца в текст и безуспешного поиска.

Определить функцию (или несколько функций) для реализации алгоритма поиска. Определить делить предусловие и постусловие.

Сформировать таблицу тестов с указанием успешного и неуспешного поиска, используя большие и небольшие по объему текст и образец, провести на её основе этап тестирования.

Оценить практическую сложность алгоритма в зависимости от длины текста и длины образца и отобразить результаты в таблицу.

1. Дано предложение, слова в котором разделены пробелами и запятыми. Распечатать те слова, которые являются обращениями других слов в этом предложении.	2. Даны две строки a и b. Требуется найти максимальную длину префикса строки a, который входит как подстрока в строку b. При этом считать, что пустая строка является подстрокой любой строки. Реализация алгоритмом Кнута-Мориса-Пратта.

2. ОТЧЁТ ПО ЗАДАНИЮ 2

2.1. Математическая модель алгоритма Кнута-Мориса-Пратта

Построение префикс функции:

J от *1* до *m (*где *m* – длинна образца*)*

$$f[i] = \begin{cases} 0\text{ , если образец не имеет собственных префиксов} \\ k\text{. где } k \text{ длинна образца } c \text{ которым идёт сравнение} \end{cases}$$

Поиск в строке (*S* – строка, *O* - образец):

Если S[i+j] != P[j], сдвигаем на f[j].

2.2. Код используемый в программе

Опишем функции, используемые в программе для решения задач.

```
std::vector<std::string> split_sentence(const std::string& sentence)
{
   std::vector<std::string> words;
   std::string word;
   std::stringstream ss(sentence);

   while (std::getline(ss, word, ' '))
   {
      std::stringstream wordStream(word);
      std::string cleanWord;

      while (std::getline(wordStream, cleanWord, ','))
      {
         if (!cleanWord.empty())
         {
            words.push_back(cleanWord);
         }
      }
   }

   return words;
}
```

Рисунок 2.1 – Функция разбиения строки по пробелам и запятым

```
std::string reverse_string(const std::string& str)
{
   return std::string(str.rbegin(), str.rend());
}
```

Рисунок 2.2 – Функция инвертирования строки

```
void find_antigrams(const std::vector<std::string>& words)
{
   std::unordered_set<std::string> wordSet(words.begin(), words.end());
   std::unordered_set<std::string> wordsAnti;

   for (const std::string& word : words)
   {
      std::string reversedWord = reverse_string(word);

      if ((wordSet.find(reversedWord) != wordSet.end() && word != reversedWord) &&
(wordsAnti.find(word)) == wordsAnti.find(reversedWord))
      {
         wordsAnti.insert(word);
         wordsAnti.insert(reversedWord);
         std::cout << '\t' << word << " - " << reversedWord << " являются антиграммами." <<
std::endl;
         wordSet.erase(reversedWord);
      }
   }
}
```

Рисунок 2.3 – Функция нахождения обращения

```
std::vector<int> prefix_function(const std::string& pattern) {
   int m = pattern.length();
   std::vector<int> pi(m, 0);
   int k = 0;

   for (int i = 1; i < m; i++) {
      while (k > 0 && pattern[k] != pattern[i]) {
         k = pi[k - 1];
      }

      if (pattern[k] == pattern[i]) {
         k++;
      }
      pi[i] = k;
   }
   return pi;
}
```

Рисунок 2.4 – Префик функция

```
std::vector<int> KMPsearch(const std::string& text, const std::string& pattern) {
    int n = text.length();
    int m = pattern.length();

    std::vector<int> pi = prefix_function(pattern);

    std::vector<int> matchPositions;
    int j = 0;
    for (int i = 0; i < n; i++) {
        while (j > 0 && text[i] != pattern[j]) {
            j = pi[j - 1];
        }

        if (text[i] == pattern[j]) {
            j++;
        }

        if (j == m) {
            matchPositions.push_back(i - m + 1);
            j = pi[j - 1];
        }
    }

    return matchPositions;
}
```

Рисунок 2.5 – Алгоритм поиска Кнута-Мориса-Пратта

```
int max_prefix(const std::string& text, const std::string& pattern) {
    std::vector<int> pi = prefix_function(pattern);
    int maxPrefixLength = 0;
    int j = 0;


    for (int i = 0; i < text.length(); i++) {
        while (j > 0 && text[i] != pattern[j]) {
            j = pi[j - 1];
        }
        if (text[i] == pattern[j]) {
            j++;
        }


        if (j > maxPrefixLength) {
            maxPrefixLength = j;
        }
    }
    return maxPrefixLength;
}       delete[] table;
        this->table = newTable;
        this->size = new_size;
    }
```

Рисунок 2.6 – Функция нахождения самого длинного префикса

2.3. Результаты тестирования

Протестируем функции, используемые в первом задании.

```
Изначальная строка: А нос упал в сон, кот побежал к току, куст пошел к туку, а мир обернулся в рим и лад обратился в дал
Антиграммы:
        нос - сон являются антиграммами.
        мир - рим являются антиграммами.
        лад - дал являются антиграммами.
```

Рисунок 2.7 – Тестирование функций

Протестируем функции, используемые во втором задании.

```
text: bcabcabccbgabddabcd
pattern: abcd
Максимальная длина префикса , которая входит в текст: 4
```

Рисунок 2.8 – Тестирование функций

4.ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Лекции по Структуры и алгоритмы обработки данных / Рысин М. Л. Москва, МИРЭА — Российский технологический университет.

2. Материалы по дисциплине Структуры и алгоритмы обработки данных / Скворцова Л. А. Москва, МИРЭА — Российский технологический университет.